UpTime

The newsletter for:
RS-DOS, OS9, OSK,
CoCos, and 68xxx's.

Volume 2, Issue 11 July 1994

Editorial

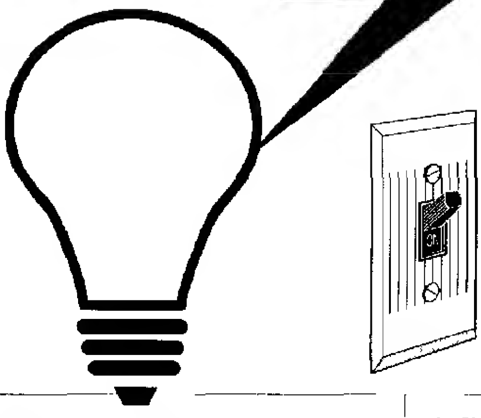
How many times have you written a program to help you out with a certain task? Sometimes these range from small speciallized calculation programs to elaborate graphic editing programs that you designed to create the "dudes" for your new action-adventure video game? Granted that everybody isn't writing the next Nintendo hit, but most of us have written a neat little utility from time to time.

So what should you do with it? Well, you could keep it to yourself, but that won't do anybody else any good. And just as you've used other programs for you own benefit, perhaps others may be able to benefit from your programming talents! A first step is to show your program at a local club meeting to anybody who may seem interested... give them a copy and they just might take it home and use it. If you're even more ambition, you may want to upload your creation to one of the many online services such as CompuServe, Delphi, and GEnie, which each have CoCo-specific areas. If you have Internet access, there are also a number of FTP servers that carry some CoCo files.

And what happens if you've created a smashing hit? Well, write to your favorite vendor, include a copy of your program, and ask if they would consider marketing it. For a commission, they will advertise and distribute your program... and you would receive the royalties!

-Jordan Tsvetkoff

Turn on your creativity!



Your next creation might benefit other CoCoists!

The Dangers of Orphanware

Disclaimer: I am not a lawyer. I do not pretend to be one. Any information provided in this article is for informative purposes only.

Many great CoCo programmers have long since moved on to other things leaving behind a legacy of software which is no longer supported. Sometimes another vendor will pick up these items and continue to market them. More often than not, though, the products just vanish.

What happens to these orphaned programs long after their creators have left them? It seems a real shame to let them go to waste. It is this line of thought which brings along the concept of "orphanware", distributing products which can no longer be purchased. This definition goes hand in hand with thinking that you are only "pirating" if the software is still being sold.

Rationalizing piracy is as simple as this: If there is no way to legally buy the item, who are you hurting if you just grab a copy from somewhere? By definition, copyright means the RIGHT to COPY. The owner of the copyright is the only one allowed to copy that item (or give permission for it to be copied). Legally, you are violating the law if you copy a program even if there is no way to buy it.

Let's take a step further. Perhaps an author simply wants to "leave the scene", maybe even with bitter feelings towards the community due to lack of sales of something he/she put a great deal of effort in (perhaps due in part to piracy in the first place). This author may not want his code shared after that. He may not even have a reason, just as a former musician may withhold the rights to older albums which are no longer available "because he wants to". No matter what the situation is, the author still owns his program. Period. Unless released into the public domain, it is illegal for anyone to copy it. This is the law.

Now, in most cases, it is quite likely that the original author may not really care what happens to old CoCo programs he once wrote. If this is the case, legal or not, the concept of "orphanware" might actually be tolerated since, if contacted, the author would probably say "no problem" anyway. No one is getting hurt. Or are

UpTime

Editor: Jordan Tsvetkoff

UpTime is published 12 times a year by **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512. Yearly subscription rates are $15.00, or $7.50 in two installments. Foreign orders (except Canada and Mexico) add $7.00 postage. Copyright © 1994 by JWT Enterprises. Quotation not permitted. Material may not be reproduced in whole or in part in any form whatsoever without the express written permission of JWT Enterprises.

Postmaster: Send address changes to *UpTime*, **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512. *Readers:* Direct all subscription inquiries, payments, and changes of address to *UpTime Subscriptions*, **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512.

Purpose: To provide information about products, services, and activities relating to the **Tandy Color Computer** and **OSK/68xxx-based machines.**

Submissions: Submissions of any form can be sent to *UpTime Submissions*, **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512. Please note that article/editorial submissions are preferred, although short program listings will also be considered. If reimbusment is requested, please indicate in writing with the initial submission.

To order a subscription to UpTime, please send a U.S. check or money order for $7.50 to:

UpTime, **JWT Enterprises**
5755 Lockwood Blvd.
Youngstown, OH 44512.

You will be billed for the second installment of $7.50. Canadian orders, $9.00 installments. Foreign orders, $11.00 installments. Ohio residents add 6% sales tax.

they?

Suppose a "long-dead" CoCo program were suddenly to appear on a club library disk as orphanware. A short time down the line, suppose a vendor gets to thinking about this old program and decides to contact the author about distributing it. The author grants permission, the deal is made, and soon the product is available on the market again. If a few dozen people who had a need for such an item already found and purchased it on a club disk, what would that do to the current market? Sure, some (many?) would go ahead and purchase it for moral reasons or even just to get a manual, but what about the rest?

It is a case like this where orphanware could hurt someone. A recent Internet cocolist message chain discussed a rumor concerning a similar occurrence. In this case, a club reportedly put a program on one of it's orphanware disks without knowing that the product was - and has been - available from a vendor. Since no advertisements had been seen for this product since the days of Rainbow magazine, and since no one easily knew who sold the product, it was apparently assumed that it was no longer being supported. This was not the case. The product had been sold and supported at virtually all of the CoCoFests over the past several years by a group which attended the shows as a vendor.

This made many of us think. If we don't actively advertise, does that give someone just-cause to pirate our software on orphanware disks? I think not, but this is apparently what can happen.

The solution? Clubs who wish to maintain libraries of orphanware need to aggressively try to contact the author or former distributor. In

Upcoming Events

5th Annual Atlanta CoCoFest
October 1st and 2nd, 1994
9am to 5pm Saturday,
9am to 3pm Sunday
Holiday Inn, Northlake
Resv: (800)-465-4329
or (404)-938-1026

Contact:
Atlanta Computer Society
P.O. Box 0694
Atlanta, GA 30366
BBS: (404)-636-2991

Tickets:
$10 for entire show
$35 for vendor table and one admission

many cases the author will grant permission to copy or, better yet, work out a deal for the club to market the product (and help raise funds for the group).

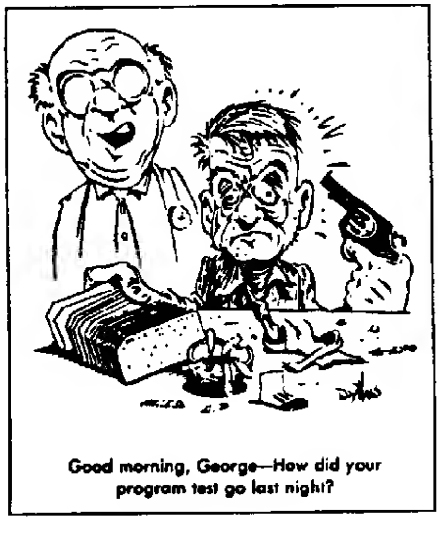
If the author cannot be located, attempts must be made with existing publications, networks, and other sources to find out if anyone else can help in the quest. If this still leads nowhere, be careful. It is up to the copier to prove they have the "right" to do so. As authors, we should do our best to document all of our software with ways to contact us... otherwise, we might take some time off and come back only to find our software collection available for $5 a disk...

It's something to think about. As a member of our Community, take a moment to think before you pick up an orphanware disk. Have you tried to find out if the program is legally available? Are you taking a sale away from a vendor, possibly bringing them one step closer to giving up on our community due to lack of sales?

We are our future. Let's continue to make it bright.

[Editor's and Author's note: The rumored discussion in the previous article for example only and should not be construed as a true story]

- Allen C. Huffman
Owner, Sub-Etha Software, *COCO-SYSOP* on GEnie, *SYSOP@DELTA* on StG Net, *COCO-SYSOP@GENIE.GEIS.COM* on Internet, or P.O. Box 152442, Lufkin, TX 75915 / (409) 560-9696 (days)



Good morning, George—How did your program test go last night?

Basic09 in ?? Easy Steps

As promised in the last installment, this time we will take a look at converting commands from DECB to Basic09. First of all, there are a number of commands in DECB that are *not* supported under Basic09, so if your program contains any of those you can start stripping them right away.

For starters, there are the commands dealing with the cassette port. The way in which the CoCo reads the cassette port demands constant attention from the microprocessor during data transfers. Because OS-9 is a multitasking operating system, it cannot pay that kind of attention to a particular application (even when you have only one program running OS-9 will interrupt it 60x per second to do its own chores).

So out go CLOAD, CLOADM, CSAVE, CSAVEM, SKIPF, AUDIO and MOTOR. If you have to you can still control the motor relay from within Basic09: POKE $FF21,56 will turn it ON, while POKE $FF21,48 turns it OFF.

The next batch of commands to go are the ones dealing with memory allocation and machine language. LPEEK and LPOKE are not supported because OS-9 partitions the computer's memory and you cannot directly access memory locations outside your own (64K) workspace.

DEFUSR, USR and VARPTR are not supported because under Basic09 you use a very different way of interacting with ML subroutines: the subroutine has a name and you call it with a RUN statement. EXEC is not supported either because under OS-9 code has to be position independent so there is little use in trying to execute code at a certain address because you don't know if there is any code there or just "garbage".

The DEFFN label is also not recognized by Basic09. You can still use the formula involved with the function but you will have to make it part of the calculations, rather than predefining it.

PLAY is not supported either, which is just as well: under OS-9 you pretty much need special drivers to do a good job with sound (unless you like the sound of machine guns).

WIDTH is not supported. If you have to set up a text window, use the gfx2 module (this was discussed in part 4) and define the window as type 1 (40 column) or type 2 (80 column).

STRING$ is another casualty. If you want to define a long string of repeating characters under Basic09, you must either type all the characters or define a short string and add it a number of times to itself using a loop.

The last command in this row is TIMER. Basic09 does provide you with a timer (DATE$) but this variable holds the current date and time. Its smallest time increment is a second, unlike TIMER which counts 60 ticks per second. (Actually OS-9 keeps time using those same ticks but they are not available to programs.)

I know that there are more commands/functions not supported by Basic09; however, rather than throwing them all on one heap, I will try to keep the story a little structured. I am currently skipping all commands that are part of the "DISK" ROM of DECB. For OS-9 a lot of these functions are part of the operating system itself and not of Basic09. This results in a lot of those functions being accessed indirectly through the SysCall utility rather than with Basic09 commands.

The same story is basically true for I/O through the serial port. Screen functions (both text and graphics) are handled through the gfx2 module (gfx for lower resolution screens) and the basics of that have been discussed in part 4 of this series.

Another big difference between Basic09 and DECB is that Basic09 has three different modes whereas under DECB you're always in the same (interactive) mode. What I mean is this: under DECB you can enter program code, edit it and execute it (or directly execute commands) all from the OK prompt.

Basic09 on the other hand has several different modes each with its own possibilities and limits. RENUM, for instance, can be use in edit mode but not in any other. When you start Basic09 you end up in the COMMAND mode.

QUALITY SOFTWARE
AT
DOWN TO EARTH PRICES

#GS01 - Pensave Graphic System
Disk Includes:
- Pen and Paint 3 Doodle Program
- Pensave-HS2 Drawing Program
- PN640-HS4 Drawing Program
- Coco Colorbook-HS2 Paint Program
- PPrint-Graphics Print Program
- Plus 6 Graphics

#GF01 - Graphic Disk for Pensave
Disk Includes.
- Penload-HS2 & HS4 Graphic Loader
- SRcolor-Loads and Paints Sunrise
- Plus Graphic Files J3CUB, J100, AL, ALT00, TWONTREE, PA1921, DOGSDAY, C7BULLET, and SUNRISE

Req: 128K Coco III & Disk Drive
$3 95 per disk or $7.95 for both
S&H $1 00 U S and Canada, $3 00 other
Check or Money Order
MO residents add 4 725% sales tax

Southern Missouri Softworks
HCR7, Box 218
Doniphan, MO 63935
(314) 255 3722

Summertime is "off" season for a lot of CoCoists. If you are one of those: look forward to new releases and upgrades this fall. If you use your CoCo all year round the following titles are currently available :

CoCoTop version 1.0	$24.95
CoCoTop version 1.1	$19.95
CoCoTop 1.1 + Tools 3	$34.95
OScopy/RScopy	$10.00
TOOLS 3 version 1.1	$29.95
Quickletter version 2.0	$19.95
Accounting level 2	$34.95
Investing level 2	$24.95
Level II graphics 1.2	$34.95

upgrades $5.00

Chris Dekker RR #4 (506) 276-4841
Centreville NB EOJ 1HO Canada

shipping+handling US/Canada $3.00 others $5.00
Prices in US dollars Canadian dollar prices: call
Send cheque or money order NO COD's
Mention the name of this magazine in your order and you will receive a free bonus disk.

Looking For Full and Real CoCo Support?
Look to: Mid Iowa & Country CoCo's
"The UPGRADE" Disk Magazine!

- Is not hard copy, though it will dump to your printer
- Displays first rate 16 color H2 graphics, with articles.
- Does keep you informed with news from around the country.
- Includes ads and recommendations of better dealers.
- Does have OS-9 articles concerning the CoCo. Has a Level II tutorial series and others. with extensive OS-9 library.
- Does have Basic programing technique article and tutorials.
- Does have a full variety of RS-DOS & OS-9 indepth articles and reviews with tips for all; where to buy printers, drives, and hardware
- Does have a free buy/sell CoCo hardware listing. With CoCos and drives plus the hard to find items.
- Some are double disk issues!

This is a News disk, not a software disk. That we leave to the "Mid Iowa & Country CoCo" library "available wherever you are at". You can select from a variety of the *best* available RS & OS Public Domain, Shareware, and Orphanware for only a filing, backup, and mailing fee. Plus a Christian software sub-chapter. ROM burn and other support to "all" members. Gosh what don't we offer? Bonus disks? Yes, we've had a bonus disk or two per year too!

Now in our ninth year, four as a national Diskmagazine! With "The UPGRADE National Diskmagazine", we've grown to be one of the largest CoCo outreaches! With over 250 satisfied subscribers in over 40 states and 5 provinces of Canada; plus Australia and England. Join with those who write: "*Here's my renewal, I don't want to miss an issue!*"
May we also serve you?

Your UPGRADE subscription includes:
• 1-year membership in MI&CC and • UPGRADE Diskmagazine subscription with
• bonus disks and double issues (Req: 128K CC3, w/1 drive, RGB or TV)
$16.00 US • $21 Canada • $31 Foreign Air

Say, "I saw it in **UpTime**" and receive,,, an UPGRADE Disk plus added bonus disk via return mail!

Terry Simons (MI&CC Treas.), 1328 48th St.
Des Moines, IA. 50311 (515) 279-2576 after 8:00 PM
Include your Phone & System information

JWT Enterprises

UpTime

A monthly magazine for CoCo's and OSK machines. **In each issue:** Find out about new products and upgrades for your favorite computers. ☞ Learn about what new technical breakthroughs are on the horizon. ☞ Discover what programs really can do and what they cannot. ☞ Published monthly in newsletter format. *One Year Subscription, two installments of $7.50; Canadian Orders, two installments of $9.00; Foreign Orders, 2 X $11.00 Can also be paid in one installment*

UpTime Back-Issues

From September 1992. Limited supplies. *$1.50 each; Foreign Orders, add $1.25 each; Bulk orders of six or more, $1.25 each; Foreign Orders add $1.00 each*

UpTime Volume 1 Pack

All 12 issues of UpTime's first year of publication are contained in a three-ring folder for their protection. If you missed them the first time around, now is your last chance as there are only a limited number available. September 1992 - August 1993. *Volume One Pack, $18.00; Canadian Orders, $9.00; Foreign Orders, $11.00*

[illegible]
(216)-758-7694

JWT Enterprises
5755 Lockwood Blvd
Youngstown, OH 44512

RAINBOW

Foreign postage excludes U.S. Territories and Canada. These products for OS-9 Level 2 on the CoCo 3. Sorry, no C.O.D.'s or credit cards, Foreign & Canadian orders, *please* use U.S. money orders U.S. checks, allow 4 weeks for receipt of order. Ohio residents, please add 6% sales tax.

Here you can use commands like MEM, EDIT, RUN, DIR, etc. (see page 10-9 for a complete list). As you can see a command like NEW is not supported but typing KILL* will have exactly the same effect. Note you are in the command mode whenever your prompt reads 'B:'.

Basic09 also has an EDIT mode (page 10-10; prompt is E:) that you can enter from the command mode. In this mode you enter your program code and Basic09 also checks your program for errors. Unlike DECB you do not automatically exit this mode after a line is edited.

A third mode is the DEBUG mode (page 10-11; prompt is D:). This mode is automatically entered whenever a running program runs into an error, unless you trapped that particular error. Two other ways of entering the debug mode are inserting a PAUSE instruction in the program or pressing <CTRL><C> from the keyboard when a program is running.

Note that this mode is only for getting the final bugs out of a program. As long as the edit mode reports errors in your program you cannot start it. Also, once your program is PACKed it can no longer enter the debug mode when it encounters an error. If you have not dealt with that error in the program it will simply crash. (I suppose you could say running packed programs is a fourth mode.)

Now back to conversions. Fortunately there are also a considerable number of commands that do the same job under both languages. Some of them control program flow and most mathematical functions also fall into this category. Here's a list of them: READ/DATA, END, (ON..) GOSUB, (ON..) GOTO, INPUT, LET, POKE, REM, RESTORE, RETURN, RUN, STOP. I suppose I could add DIM also to this list but keep in mind that you cannot dynamically DIMension variables as I mentioned in part 5.

A list of functions that need no further attention: ABS, ASC, ATN, CHR$, COS, EXP, FIX, INT, LEFT$, LEN, LOG, PEEK, RIGHT$, RND, SGN, SIN, STR$, SQR, TAN, VAL. All operators (+,-,>,=,etc.) fall in this category as well.

If you want to port over an entire program without retyping it, your best bet is to make a copy of the program in an ASCII file (save with ,A option) and, after copying it to an OS-9 disk, first use OS-9's edit command or wordprocessor to do some preprocessing.

The reason for this is fairly simple: whenever you make a change in a line using Basic09's editor, that line will immediately be checked for errors. If one is found the editor aborts the command you gave it and points out that error to you.

Normally this works great, but when porting a program this may be less desirable. For instance, if you use more than one statement per line DECB uses ':' as a separator. Basic09 on the other hand uses '\'. This is no big deal if you can replace them with one global replace command. On the other hand, if you have to replace them one by one it becomes very frustrating. Since OS-9's edit command acts more like a word processor and doesn't perform error checking, it is more suited for the task.

And now for the hard (or frustrating) part: dealing with the commands that look oh so familiar, but behave just a little different. The first one is CLOSE. Under DECB this statement has a number (channel) attached to it while under Basic09 it has a variable attached to it. This variable has been defined earlier in the program (by an OPEN statement for instance). If you got that working you shouldn't have too much trouble with CLOSE.

IF .. THEN .. ELSE has the same control function under Basic09. It is just a little pickier so keep the following things in mind: *always* terminate the block with an ENDIF statement because Basic09 does not see the end of a line as an implied endif. Basic09 accepts the ELSE statement only if it is the first word on a new line *or* if it directly follows a backslash (\). If your IF .. THEN statement points to a line number you *must* omit the ENDIF part otherwise Basic09 generates an error.

For INPUT# the same rules apply as for CLOSE.

With MID$ we are back to the old days: under Basic09 this command behaves in the same way as it did under Color Basic (before Extended Basic came along). It simply returns a portion of the target string rather than replacing it with a new string. The syntax, however, is exactly the same as for the DECB command.

Actually replacing a piece of a string under Basic09 is a rather tedious operation: you will have to split the original using left$ and/or right$ and then build a new string by concatenating the various pieces: A$=B$+C$+D$ etc.

A FOR..NEXT loop is much the same as in DECB, but you have to keep in mind that you *must* specify a variable in your NEXT statement otherwise Basic09 will report an error. Another thing is that Basic09 wants to see one NEXT per loop. Terminating multiple loops (e.g. NEXT I,J,K) with one statement doesn't work.

ON ERR GOTO must be changed a little bit into ON ERROR GOTO, but has the same function in both languages.

The OPEN statement has more or less the same function in Basic09 but a very different syntax. In its generic form it looks like OPEN #path,filename:accessmode. For your program to work, "path" must be a byte or integer type variable. Your program can later use that variable to access the file but it cannot assign a value to the variable. OS-9 will do this for you and if you change its value you will no longer be able to access the file. "Filename" can be a (string)variable or a literal string. If you use the latter option, you must enclose the name in quotation marks. "Accessmode" under Basic09 is not defined with a letter but with a word. Most commonly used are READ, WRITE, and UPDATE (which allows reading and writing to a file). Two more options are DIR and EXEC. DIR allows you to access directory files, while EXEC causes Basic09 to work with the execution directory as opposed to the data directory. You may specify more than one accessmode if necessary.

PRINT behaves mostly in the same way as under DECB except the PRINT # statement which is generally used with a variable (i.e. PRINT #path). OS-9 reserves the three lowest pathnumbers (0,1,2) of every process for input, output and error paths. So if you use the statement PRINT #1,"A" you will see a capital A appear on the screen. By default the error path (#2) is also connected to the screen and the input path (#0) is connected with the keyboard. Since all characters typed are echoed to the screen, PRINT #0, PRINT

#1 and PRINT #2 have the same effect until you redirect the input and/or output paths. You do so with the <, > and >> modifiers on the OS-9 commandline. In those cases your message will be printed to whatever device you choose: another window, a printer, diskfile, etc.

PRINT @ is not supported by Basic09. For positioning text on the screen see the description of LOCATE.

PRINT USING under Basic09 is much more comprehensive than with DECB. Starting at page 11-122, Basic09's manual devotes 7 pages to explaining it and I don't feel like repeating that over here. A few quick points will do:

Basic09 recognizes 6 formats: string, boolean and 4 number types. For most formats (including strings) you can use the <, > and ^ modifiers to get things in place. Unlike DECB you do not repeat a character but use a number-symbol combination to tell Basic09 the size of the field it can use. For instance ##### would become I5 if the value is an integer or R5.0 if it is a real number, while %% would be replaced by S6. Note that Basic09 also recognizes Txx for tabulating and Xxx for inserting spaces. The entire string of symbols must be enclosed in quotation marks, followed by the various variables that have to be printed out. For instance PRINT USING "x2,s10<,R7.2^","Total",amount would be printed in this fashion:

```
Total     100.00 (ifamount=100)
```

If you use the tabulation functions [TAB(xx) or Txx] keep in mind that they behave differently if output is send to the printer as compared to the screen. For instance the line PRINT #path, TAB(25);filename will print a filename starting at column 25 when path points to the screen. This will always work as long as the current cursor position is somewhere in columns 0-24. However if path is connected with your printer the same line always results in printing 25 spaces, regardless of the position of the print head.

GET and PUT under DECB are graphics functions. Although they have the same functions under Basic09 when used in conjunction with the Gfx2 module [i.e. RUN Gfx2("get",...)] they also have a very different function in the form of the Basic09 GET and PUT statements. In that case they are simple I/O functions like INPUT and PRINT.

There is one big difference between GET/PUT and INPUT/PRINT: while the latter two will format the data they deal with, GET/PUT will just transfer it. For instance if you type DIM buffer(500):BYTE and then GET #path, buffer GET will (attempt to) read 500 bytes via that path. Suppose that path is associated with a file. GET now reads the first 500 bytes after the filepointer regardless of the meaning of the data. It could be 250 integers, 100 real numbers or 10 50-character strings; that just makes no difference at all. PUT will perform the same way but writes data to a path.

These commands are mostly used with complex data structures where you can mix various types of variables in one record. With GET/PUT you can read/write the entire record (or array for that matter) with one command. Be careful when using GET for input from the keyboard. For example if your program has the instruction GET #0, key but does not define "key"; your computer will "hang up" until you have pressed 5 keys. The reason for this is that Basic09 regards every undefined variable as a REAL number which is made up of 5 bytes in memory. So the instruction insists on reading 5 bytes before returning control to Basic09.

INKEY$ is not directly supported by Basic09, but shipped as a separate module. Consequently a statement like A$=INKEY$ translates into RUN INKEY(A$). Make sure the INKEY module is available to OS-9. It is generally a good idea to merge it with Basic09 and/or the RunB module into one file.

LOCATE is not handled by Basic09 either but by the Gfx2 module and more specifically its CURXY function. LOCATE 20,12 translates into RUN Gfx2("curxy",20,12).

INSTR has an equivalent under Basic09 that is called SUBSTR. There is one big difference between the two: with SUBSTR you cannot specify a starting column. If your program is absolutely dependent on that your can translate A=INSTR(5,search$,"find") into the following code:

```
temp=right$(search$,len(search$)-
     5)
A=substr("find",temp)
```

Note that with SUBSTR, the search and target string are reversed as compared to INSTR. SUBSTR will return 0 if it cannot find a match in temp.

EOF returns TRUE or FALSE under both languages, but its syntax is slightly different under Basic09. Typically one uses the statement IF EOF(#path) THEN with Basic09 to determine whether the end of a file has been reached. Note that in this case "path" refers to the variable used in the OPEN statement.

The JOYSTK function is not supported under Basic09. That doesn't mean you cannot read the joystick ports— it just means it takes a little extra work to do it. Under OS-9 we have to read them through a system call. First you must set up a data structure representing the 6809's registers. How to do this was shown in an earlier part of this series.

To execute this particular call use the following code:

```
regs.a=1 \ regs.b=$13
regs.x=0 (left stick; use 1
    for right)
run SysCall($8D,regs)
```

Register A returns the button status (a code ranging from 0-3); while regs.x returns the x coordinate and regs.y returns the y coordinate of the joystick's position. For more info on this call see page 8-113 of the technical reference section of your OS-9 manual.

Error trapping is basically handled in the same way under both languages, but the various functions have different names. I already discussed ON ERROR GOTO. ERLIN is not supported because, by default, Basic09 uses no line numbers. The ERNO function is called ERR under Basic09. Your basic error trapping routine would look something like this:

```
ON ERROR GOTO 100
100 errnum=ERR
```

Note that we must catch the value

Announcing

5th Annual *Atlanta* CoCofest

Holiday Inn, Northlake

October 1 & 2, 1994

Show Hours:

Sat. Oct 1	9:00 AM - 5:00 PM
Sun. Oct 2	9:00 AM - 3:00 PM

Vendor setup:

Fri. Sept 30	6.00 PM - 9:00 PM
Sat. Oct 1	8:00 AM - 8:45 AM

Admission: $10.00. (Whole Show)

Reservations:

Holiday Inn, Northlake
(800)-465-4329 or (404)-938-1026

Sponsored by:

Atlanta Computer Society
PO Box 80694
Atlanta, GA 30366
BBS: (404)-636-2991

'I Must Warn You, Mr. Munari, Its Memory Is Advanced Enough to Never Forget Your Refusal to Buy It Right Now.'

held in ERR in a variable because Basic09 resets its value to 0 as soon as you read it. If an error occurs in this example ERR will hold the error code, while program execution jumps to line 100 and continues from there.

The ON BRK GOTO statement is not supported under Basic09 either. However since pressing the BREAK key will invariably get you an error 2; you can replace this statement by IF errnum=2 THEN sometime after line 100 (in the error trapping routine). Note that you cannot test this feature if you run your programs from within Basic09 because Basic09 traps this error and interrupts the program rather than passing on the error code to it. However once you have PACKed your program and run it using the RunB module it will work.

The last thing I want to mention is, that Basic09 supports an extra command called ERROR. Whenever Basic09 encounters this command it generates an error and then jumps to the error trapping routine to deal with the error. This comes in handy for debugging a program or when your program has "dead ends" in it. You can terminate that piece of code with an ERROR statement followed by a number and deal with the situation in your error trapping routine.

Basic09 has a lot more commands that are not supported by DECB but since you don't really need them when converting a DECB program I won't deal with them here. Next time we'll take a look at the commands inside the "DISK" ROM of DECB.

- *Chris Dekker*

Advertiser's Index

Atlanta	7
ColorSystems	7
Dekker	4
Hawksoft	2
JWT Enterprises	4
Mid Iowa & Country C'b	4
Southern Missouri	4

UpTime
JWT Enterprises
5755 Lockwood Blvd.
Youngstown, OH 44512

Address correction requested